Using Artificial Intelligence Planning to Automate Science Image Data Analysis

Steve Chien, Forest Fisher, Edisanter Lo, Helen Mortensen, Ronald Greeley

Steve Chien, Forest Fisher, and Helen Mortensen are affiliated with the
Jet Propulsion Laboratory,
California Institute of Technology,
Pasadena, CA 91109-8099

Edisanter Lo and Ronald Greeley are affiliated with the
Department of Geology,
Arizona State University,
Tempe, AZ 85287-1404

Contact Author:
Steve Chien
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, M/S 126-347
Pasadena, CA 91109-8099
steve.chien@jpl.nasa.gov
(818) 393-5320 (Voice)
(818) 393-5244 (FAX)

Keywords: planning, reconfiguration of software modules, science data analysis

Abstract

In recent times, improvements in imaging technology have made available an incredible array of information in image format. While powerful and sophisticated image processing software tools are available to prepare and analyze the data, these tools are complex and cumbersome, requiring significant expertise to properly operate. Thus, in order to extract (e.g., mine or analyze) useful information from the data, a user (in our case a scientist) often must possess both significant science and image processing expertise.

This article is an extended version of [8] and describes the use of AI planning techniques to represent scientific, image processing, and software tool knowledge to automate knowledge discovery and data mining (e.g., science data analysis) of large image databases. In particular, we describe two fielded systems. The Multimission VICAR Planner (MVP) which has been deployed for since 1995 years and is currently supporting science product generation for the Galileo mission. MVP has reduced time to fill certain classes of requests from 4 hours to 15 minutes. The Automated SAR Image Processing system (ASIP) was deployed at the Dept. of Geology at Arizona State University in 1997 to support aeolian science analysis of synthetic aperture radar images. ASIP reduces the number of manual inputs in science product generation by 10-fold.

Introduction

Recent breakthroughs in imaging technology have led to an explosion of available data in image format. However, these advances in imaging technology have brought with them a commensurate increase in the complexity of image processing and analysis technology. When a scientist analyzes newly available image data to discover patterns or to confirm scientific theories, they must perform a complex set of operations. First, before the data can be used it must often be reformatted, cleaned, and many correction steps must be applied. Then, in order to perform the actual data analysis, the user must manage all of the analysis software packages and their requirements on format, required information, etc.

Furthermore, this data analysis process is not a one-shot process (indeed, we describe a specific case of the process outlined in [13]. Typically a scientist will set up some sort of analysis, study the results, and then use the results of this analysis to modify the analysis to improve it. This cycle of repeated analysis may occur many times - thus any reduction in the scientist effort or cycle time can dramatically improve scientist productivity.

Unfortunately, this data preparation and analysis process is both knowledge and labor intensive. Consider the task of producing a mosaic of images of the moon from the Galileo mission (corrected for lighting, transmission errors, and camera distortions). Consider also that our end goal is to perform geological analyses - i.e., to study the composition of the surface materials on the moon. One technique used to do this is to construct a ratio image - an image whose values are the ratio of the intensity of the response at two different bandwidths (e.g., the ratio of infra-red response and visible green response). In order to correctly produce this science product for analysis, one must have knowledge of a wide range of sources including:

- the particular science discipline of interest (e.g., atmospheric science, planetary geology),
- image processing and the image processing libraries available,
- where and how the images and associated information are stored (e.g., calibration files), and
- the overall image processing environment (e.g., to know how to link together libraries and pass information from one program to another).

Note the extreme breadth of knowledge required to perform this task - it requires science, image processing, database infrastructure, and image processing language and scripting programming knowledge. As a result of the vast amounts and breadth of knowledge required, it takes many years of training and experience to become expert at assisting I these analyses.

Automated planning technology offers the potential automate many of these data analysis functions [13(page 50), 5], thus enabling novice users to utilize the software libraries to mine the data. It also allows users who may be expert in some areas but less knowledgable in other to use the software tools to mine the data.

The remainder of this article is organized as follows. First, we provide a brief overview of the key elements of AI planning. We then describe two fielded planning systems for science data analysis. We first describe the MVP system - which automates elements of image processing for science data analysis for data from the Galileo mission. We then describe the ASIP system - which automates elements of image processing for science data analysis of synthetic aperture radar (SAR) images.

The principle contributions of this article are twofold.

- First, we identify automated selection and configuration of KDD software tools as an area where AI planning technology can significantly extend KDD capabilities.
- Second, we describe two systems demonstrating the viability and impact of AI planning on the KDD process[1].

Artificial Intelligence Planning Techniques

We have applied and extended techniques from Artificial Intelligence Planning to address the knowledge-based software reconfiguration problem in general [10], and two applications in science data analysis (e.g., data mining) in specific. In order to describe this work, we first provide a brief overview of the key concepts from planning technology[2].

Planning technology relies on an encoding of possible actions in the domain. . In this encoding, one specifies for each action in the domain: preconditions, postconditions , and subactivities . Preconditions are requirements that must be met before the action can be taken. These may be pieces of information that are required to correctly apply a software package (such as the image format, availability of calibration data,

[1] For a description of other planning and scheduling work at the Jet Propulsion Laboratory (JPL) see [9, 6]
[2] For further details on planning the user is referred to [26], [12], [29], [30]

etc.). Postconditions are things that are made true by the execution of the actions, such as the fact that the data has been photometrically corrected (corrected for the relative location of the lighting source) or that 3-dimensional topography information has been extracted from an image. Substeps are lower level activities that comprise the higher level activity. Given this encoding of actions, a planner is able to solve individual problems, where each problem is a current state and a set of goals. The planner uses its action models to synthesize a plan (a set of actions) to achieve the goals from the current state.

Planning consists of three main mechanisms: subgoaling, task decomposition, and conflict analysis. In subgoaling, a planner ensures that all of the preconditions of actions in the plan are met. This can be done by ensuring that they are true in the initial state or by adding appropriate actions to the plan. In task decomposition, the planner ensures that all high level (abstract) activities are expanded so that the lower level (subactivities) are present in the plan. This ensures that the plan consists of executable activities. Conflict analysis ensures that different portions of the plan do not have negative interactions that will cause the plan to fail.

AI planning researchers have developed numerous approaches to the task of correct and efficient planning. Two main planning methods are operator-based planning and hierarchical task network (HTN) planning. Our work in developing automated planning systems for science data analysis uses a combination of both these approaches, exploiting the advantages of each. HTN planning most naturally represents task decomposition. Operator-based planning most naturally represents subgoaling. Conflict analysis is relevant to both task decomposition and subgoaling, as both:

1. the choices one makes to expand a higher level activity into a set of lower level activities; and
2. the choices one makes when finding an action to satisfy a precondition for another action are constrained by the problem of interference with other portions of the plan.

Both HTN and operator-based planners typically construct plans by searching through a space of potential plans (called a plan-space). However, they differ considerably in how they search. HTN planners specify plan modifications in terms of flexible task reduction rules and work in a forward-chaining, top-down fashion. In contrast, operator-based planners work in a backward-chaining manner by taking a given goal and attempting to resolve its preconditions. Operator-based planners perform all reasoning at the lowest level of abstraction and provide a strict semantics for defining operator definitions.

An HTN planner [7] uses task reduction rules to decompose abstract goals into lower level tasks. HTN planners can encode many different types of information into task reductions. By defining or not defining certain reduction refinements, the designer can direct the planner towards particular search paths in certain contexts. The user can also directly influence the planner by explicitly adding an ordering constraint or goal protection that would not strictly be derived from goal interaction analyses. Search-control knowledge can also be encoded by writing explicit action sequences to achieve goals, thereby avoiding considerable search.

In contrast, an operator-based planner [26, 29] reasons at a single level of abstraction -- the lowest level. Actions are strictly defined in terms of preconditions and effects. Plans are produced through subgoaling and goal interaction analyses. In this framework, all plan constraints (protections, ordering, and codesignation) are a direct consequence of goal achievement and action precondition and effect analysis. Thus, an operator-based planner generally has a strict semantics grounded in explicit state representation, i.e. defining what is and is not true in a particular state (or partial state).

Our approach to planning (as embodied by the DPLAN planner [7] combines these two planning methods, utilizing the advantages of each. For instance, an operator-based planner requires a very rigid representation. This is both a strength and a weakness. It is an advantage in that there is usually one obvious method of encoding each subproblem. However this rigidity can also make certain aspects of a problem difficult to represent. Known ordering constraints and operator sequences can be difficult to encode if they cannot easily be represented in terms of preconditions and effects. Such constraints can and are often forced by adding dummy preconditions - in which an operator A is made to precede an operator B by forcing A to achieve a condition C for B. However this solution can often create a misleading representation in that other occurrences of A do not require C to be true. An HTN planner, on the other

hand, allows the easy representation of known ordering constraints. Domain information, such as constraints, is easily added to domain rules in the HTN framework. This type of representation allows the user to easily direct the planner's search by explicitly defining items such as ordering constraints and goal protections.

By using a combination of both HTN planning and operator-based planning we can easily direct search and can define knowledge in an understandable top-down fashion. In a hybrid representation we also have the ability to define knowledge in the more structured operator-based fashion when appropriate.

DPLANs algorithm is a combination of both hierarchical task network (HTN) planning techniques and operator-based planning techniques. In HTN planning, abstract actions such as "calibrate receiver" or "configure sequential ranging assembly" are decomposed into specific directives for specific hardware types. In operator-based planning, requirements of specific actions, such as "move antenna to point", are satisfied using means-end analysis, which matches action preconditions to effects and resolves any occurring ordering conflicts.

In order to apply planning technology to an application domain, one must construct a planning knowledge base that encodes knowledge of the domain. This involves encoding the goals, operators, task decompositions, and probably customizing a search strategy for the domain. These elements would populate a planning knowledge base that would then be used by a general planning engine to solve planning problems in the domain.

The DPLAN Planning Algorithm

The DPLAN planning algorithm uses a unique combination of the HTN and operator-based planning techniques discussed above. DPLAN operates by refining a set of input top-level goals into a set of low-level operational goals (e.g., executable actions). Plans are represented by a three-tuple: <U,C,S> where U is a set of non-operational (or high-level) goals, C is a set of constraints, and S is a set of operational-goals. At the end of planning, U should be empty and the goals in S are returned as the final plan steps.

For example, in the image processing case described above, each of the elements of the abstract plan would be as follows:

- the input non-operational goals (U) would be the image processing goals from the user, such as: radiometric correction, fill in missing lines, etc. ; and
- the output operational goals (S) (executable activities) would be activities representing parameterized invocations of image processing programs; and
- the constraints from a final plan (C) would specify how the outputs of one program were to be used as the inputs for other programs and the required sequence of execution.

An overview of the DPLAN algorithm is shown in Figure 1. The main inputs to DPLAN are: a set of high-level goals G, a set of decomposition rules R, and the set of all possible operational goals O. Search is implemented by keeping a queue of partial plans to be explored. Currently, plans are selected from the queue using a best-first heuristic; however, other search techniques could easily be employed. Step 1 and Step 2 of the main loop remove the best plan off the queue, and Step 3 checks if that plan is a solution. If no solution has been found then a new goal is selected for refinement in Step 4. Step 5 chooses a refinement strategy for that goal, and in Step 6, any new plans created through that strategy are inserted into the plan queue.

A plan is considered a solution if two conditions are true. The first is that there are no non-operational goals left to be refined. The second condition is that all context goals have been achieved or are directly achievable in the current plan. Context goals are goals that were needed for applying a decomposition rule, but are supposed to be accomplished by some other part of the plan. If all context goals have been achieved, then the plan is returned as a success.

Algorithm DPLAN(G,R,O)

Let R be a set of decomposition rules,
Q be a list of partial-plans,
P be the current plan, and

let P_0 be a plan <non-operational goals = G, constraints = {}, operational goals = {}>
Initialize the plan queue $Q := (P_0)$

While Q is not empty
(Step)

1. Select a promising plan P in Q using heuristics,
2. Remove P from Q
3. If P contains only operational-goals
   If the context goals in P are achieved, return P.
   Else goto 1.
4. Else choose a non-operational goal g from U.
5. Refine g.
6. Insert any new plans generated by refinement into Q.

Figure 1: The DPLAN Search Algorithm

DPLAN can use several different refinement strategies to handle non-operational goals. There are two main types of goals in DPLAN: activity-goals and state- goals. Activity-goals correspond to operational or non-operational activities and are usually manipulated using HTN planning techniques. Operational activity-goals are considered primitive tasks that can be directly executed. Non-operational activity-goals must be further decomposed into operational ones through HTN reduction rules. State-goals correspond to the preconditions and effects of activity-goals, and are achieved through operator-based planning. State-goals that have not yet been achieved are also considered non-operational. Figure 2 shows the procedures used for refining these two types of goals. As soon as a refinement strategy is applied to an activity-goal or state-goal, it is removed from the list of non-operational goals.

DPLAN can also use additional domain information for more efficient and flexible planning. For instance, a planning problem can specify a list of static context facts. These facts represent operational goals that are always considered to be true. Such goals are easy for DPLAN to verify during planning and can help in pruning off search branches. Other possible inputs include sets of preconditions and effects for operational activities, a set of final goals that must be true in the plan solution, and a set of initial goals that are true at

If g is an Activity-Goal,

1. Decompose : For each decomposition rule r in R which can decompose g , apply r to produce a new plan P', If all constraints in P' are consistent, then add P' to Q.
2. Simple Establishment : For each other activity-goal g' in P that can be unified with g , simple establish g using g' and produce a new plan P'. If all constraints in P' are consistent, then add P' to Q.

If g is a State-Goal,

1. Step Addition : For each activity-goal, g' that asserts g as an effect, add g' to P to produce a new plan P'. If the constraints in P' are consistent, then add P' to Q.
2. Simple Establishment : For each activity-goal g' in U that has an effect e that can be unified with g, simple establish g using e and produce a new plan P'. If all constraints in P' are consistent, then add P' to Q.

Figure 2: Goal Refinement Strategies

the beginning of planning. This information is not required for standard DPLAN operation, but can be very beneficial during planning.

The Multimission VICAR Planner (MVP)

MVP [5] partially automates the generation of image processing procedures from user requests and a knowledge-based model of VICAR image processing area using Artificial Intelligence (AI) automated planning techniques. VICAR image processing is an instance a planning problem where:

- the planning actions or operational goals/activities are VICAR image processing programs;
- the planning initial state is the current state of the image files of interest; and
- the planning input goals are the user image processing goals.

The VICAR environment (Video Image Communication and Retrieval[3].) [23] supports image processing for: JPL flight projects including VOYAGER, MAGELLAN, and GALILEO, and CASSINI; other space imaging missions such as SIR-C and LANDSAT; and numerous other applications including astronomy, earth resources, land use, biomedicine, and forensics with a total of over 100 users. VICAR allows individual processing steps (programs) to be combined into more complex image processing scripts called procedure definition files (PDFs). The primary purpose of VICAR is to enable PDFs for science analysis of image data from JPL missions.

Figure 3: Raw and Processed Image Files

An Example of MVP Usage

In order to illustrate how MVP assists in VICAR planetary image processing, we now provide a typical example of MVP usage to ground the problem and the inputs and outputs required by MVP. The three images, shown at the left of Figure 3 are of the planet Earth taken during the Galileo Earth 2 flyby in December 1992. However, many corrections and processing steps must be applied before the images can be used. First, errors in the compression and transmission of the data from the Galileo spacecraft to receivers on Earth has resulted in missing and noisy lines in the images. Line fillin and spike removals are therefore desirable. Second, the images should be map projected to correct for the spatial distortion that occurs when a spherical body is represented on a flat surface. Third, in order to combine the images, we need to compute common points between the images and overlay them appropriately. Fourth, because we are combining multiple images taken with different camera states, the images should be radiometrically corrected before combination.

[3] This name is somewhat misleading as VICAR is used to process considerable non-video image data such as MAGELLAN synthetic aperture radar (SAR) data

display automatic nav residual error	perform manual navigation
radiometric correction	pixel spike removal
missing line fillin	uneven bit weight correction
no limbs present in images	perform automatic navigation
display automatic nav residual error	perform manual navigation
display man nav residual error	map project with parameters ...
mosaic images	smooth mosaic seams using DN

Figure 4: Example Problem Goals

MVP enables the user to input image processing goals through a graphical user interface with most of the goals as toggle buttons on the interface. A few options require entering some text, usually function parameters that will be included as literals in the appropriate place in the generated VICAR script. Figure 4 shows the processing goals input to MVP.

Using the image processing goals and its knowledge of image processing procedures, MVP constructs a plan of image processing steps to achieve the requested goal. This plan is translated into a VICAR script which, when run, performs the desired image corrections and constructs a mosaicked image of the three input files. The finished result of the image processing task is shown at the right in Figure 3. The three original images now appear as a single mosaicked image, map projected with missing and corrupted lines filled in.

To further continue this example, shown in Figure 5 is a code fragment to perform portions of image navigation[4] for a Galileo image[5]. The higher-level conceptual steps (i.e., plan steps) are shown at the left and the corresponding VICAR code is shown at the right. In this case the overall user goal is to navigate the image. The other subgoals (and steps) are necessary to support this goal.

Thus MVP allows the user to go directly from high level image processing goals to an executable image processing program. This enhances productivity because the user can focus on which processing goals are needed for the science rather than being bogged down in details such as file format, normalizing images, etc.

MVP does not always fully automate a planetary imaging task. In typical usage, the analyst receives a request, determines which goals are required to fill the request, and runs MVP to generate a VICAR script. The analyst then runs this script and then visually inspects the produced image(s) to verify that the script has properly satisfied the request. In most cases, upon inspection, the analyst determines that some parameters need to be modified subjectively or goals reconsidered in context. This process typically continues several iterations until the analyst is satisfied with the image product.

Task Reduction in MVP

MVP represents VICAR processing and science data analysis knowledge in the form of task reduction rules [17]. For example, Figure 6 shows a decomposition rule for the problem class *mosaicking* with absolute navigation . This rule states that if mosaicking is a goal of the problem and an initial problem decomposition has not yet been made, then the initial problem decomposition should be into the

[4] Image navigation is the process of determining the matrix transformation to map from the 2-dimensional (line, sample) coordinate space of an image to a 3-dimensional coordinate space using information on the relative position of the imaging device (spacecraft position) and a model of the target being imaged (e.g., the planetary body)

[5] This code was generated by MVP.

subproblems: local correction, navigation, registration, mosaicking, and touch-ups and that these subproblems must be solved in that order.

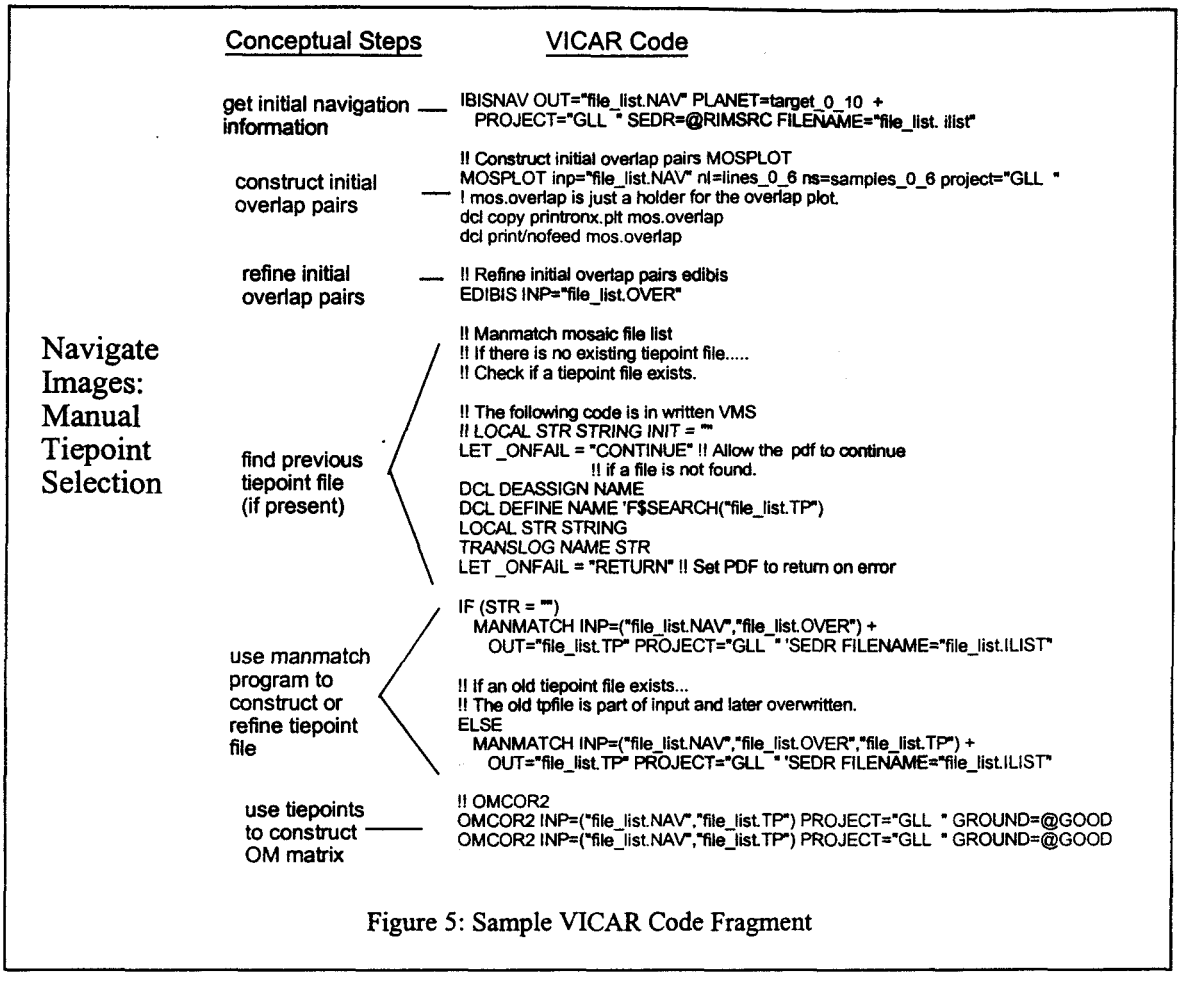

Figure 5: Sample VICAR Code Fragment

	LHS
G_L =	mosaicking goal present
C_0 =	null
C_2 =	an initial classification has not yet been made

RHS

G_R = {
1. Local correction,
2. Navigation,
3. Registration,
4. Mosaicking,
5. Touch-ups

}

C_1 = {
- these subtasks be performed in order 1.2.3.4.5
- protect local correction until mosaicking

}

N = problem class is mosaicking

Figure 6: A Task Reduction Rule from MVP

operator GALSOS

:parameters ?infile ?ubwc ?calc

:preconditions the project of ?infile must be galileo
the data in ?infile must be raw data values

:effects reseaus are not intact for ?infile
the data in ?infile is not raw data values
missing lines are not filled in for ?infile
?infile is radiometrically corrected
the image format for ?infile is halfword
?infile has blemishes-removed
if (UBWC option is selected) then ?infile is uneven bit weight corrected
if (CALC option is selected) then ?infile has entropy values calculated

Figure 7: Sample Operator

Operator-based Planning in MVP

MVP also uses operator-based planning techniques [26]. An operator-based planner uses models of actions in a domain to achieve goals from an initial world state. In the VICAR domain the actions (operators) are image processing steps, initial state the initial image file state, and the goals the processing request.

In operator-based planning, an action is represented in terms of its preconditions (required to be true before an action can be executed), and its effects (true after an action is executed). For example, the GALSOS program to radiometrically correct Galileo image files is represented Figure 7.

When constructing a plan to achieve a goal G_1, a planner will consider those actions that have G_1 as an effect (thus considering GALSOS to achieve a radiometric correction goal). In order to use GALSOS, MVP must also ensure that the preconditions of the operator are met, in a process called subgoaling. MVP must also ensure that operators in the plan do not undo conditions required by other parts of the plan - this is performed in a process called conflict analysis[6].

One novel aspect of the VICAR domain is that considerable search in planning is not at the program selection level (which corresponds to operator selection in the planning process) but rather at the program option selection level (which corresponds to selecting the appropriate operator effect after the operator has been selected). In order to efficiently handle this type of search, we have integrated a constraint reasoning mechanism which allows the planner to reason about compatible and incompatible program option settings in a least-commitment fashion (see [5] for details).

Impact of Combining Decomposition and Operator-based Planning Methods

One obvious question is the impact of combining decomposition and operator-based planning methods. Earlier in the article, we stated that the two reasons for combining decomposition and operator-based methods were user understandability and search control. While it is difficult to quantify the effectiveness of increased understandability of plans, in this section we attempt to roughly quantify the effectiveness of decomposition methods in controlling the search required by the operator-based planner.

The principle impact of decomposition planning on search is to decompose the planning process into independent subproblems that can be solved independently in a known sequential fashion. In the current MVP knowledge base, there are seven such problem spaces: local correction, automatic navigation, manual

[6] Because subgoaling and conflict analysis in operator-based planning are not unique to MVP, we have only briefly sketched their key elements. For a more detailed treatment of operator-based planning algorithms the reader is referred to [26].

navigation, photometric correction, registration, mosaicking, and touch-ups. In Figure 8 , we describe the

Problem Space	operators	goals	typical search
local correction	15	7	60
automatic navigation	20	4	150
manual navigation	24	4	300
photometric correction	5	2	60
registration	13	5	110
mosaicking	4	3	325
touch ups	10	3	325

Figure 8: Problem Space Information

salient information on each of the problem-spaces. First, for each problem space we list the number of relevant planning operators and top-level input goals as this gives some indication of the size of the problem space. We also list the typical number of plans searched in the problem space.

The overall effect of decomposition planning on search is to break down the search into more manageable subproblems. For example, if subproblem A typically requires searching α plans and subproblem B typically requires searching β plans, solving both problems simultaneously might require on order $\alpha\beta$ plans[7]. Overall, because the search spaces combine (roughly!) multiplicatively, the impact of adding domain knowledge to decompose subproblems has been enormous. For example, originally the automatic navigation and manual navigation problem spaces were represented as a single navigation problem space. However, this problem space required too much search (on the order of 50,000 plans), so it was broken into the automatic and manual navigation problem spaces.

An Example of Subgoaling in VICAR Image Processing

To illustrate how the operator-based planning process performs subgoaling, consider the subgoal graph illustrated in Figure 9.[8] In this case the user has selected the goal that the images be navigated using manual methods and that the archival navigation information for the image should be updated. The decomposition planner has access to the knowledge that in order to navigate the image, the operational goal is to construct an OM matrix which defines the transformation from (line, sample) in the image to some known frame of reference (usually the position relative to the target planet center). The planner knows that in order to compute this matrix it must have a tiepoint file, the project of the image, and the image files formatted into a mosaic file list. In order to produce a tiepoint file for the goal specification of manual navigation, the planner uses the MANMATCH program. The MANMATCH program in turn requires a refined overlap pairs file, the project of the images, the initial predict information, and again a mosaic file list. The refined overlap pairs file can be constructed using the EDIBIS program, but this requires a crude overlap pairs file based on an initial predict source. This crude overlap pairs file in turn requires the default navigation method, and the latitude and longitude of sample image files. The rest of the graph is generated similarly. This subgoal graph is generated in response to the particular combination of user goals and the state of the selected image files.

[7] This is clearly a simplification, it might require less search than this because weak heuristics might tend to guide the search well. However, it might be worse because adding the second subproblem might weaken heuristics that work well for the first subproblem alone. Empirically in the MVP image processing application combining two search spaces A and B as above would result in search slightly less than $\alpha\beta$.

[8] The VICAR code previously shown in Figure 5 is taken from this example.

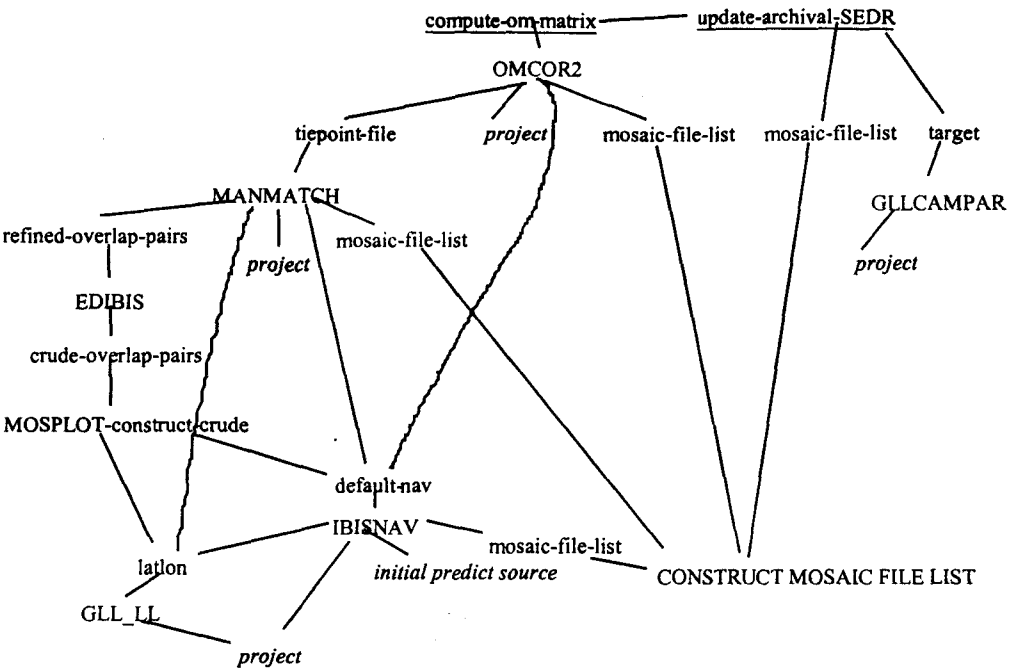

Figure 9: Subgoal graph for manual relative navigation of Galileo image files.

An Example of Resolution of Goal Conflicts in VICAR Image Processing

To illustrate how the operator-based planning process resolves interactions between steps, consider the (simplified) image processing operators shown in Figure 10. The relevant operators to achieve the goals of missing line fillin, spike removal, and radiometric correction for Voyager and Galileo images are shown below. When constructing a plan to achieve these goals, depending on the project of the image file (e.g., either Voyager or Galileo), MVP determines the correct program to use because the preconditions enforce the correct program selection.

Operator	VGRFILLIN	GLLFILLIN	ADESPIKE	FICOR77	GALSOS
Preconditions	VGR image EDR present	GLL image	GLL image or VGR image raw values	VGR image	GLL image Raw pixel values
Effects	Missing lines filled in	Missing lines filled in	Spike removal Not raw values	Blemish removal Not raw values	Reed-solomon overflow corr. Saturated pixel corr. Not missing lines filled in

Figure 10: Simplified Operator Definitions

Goal	VICAR Program		Execution Order	
	Voyager	Galileo	Voyager	Galileo
fillin missing lines	VGRFILLIN	GLLFILLIN	VGRFILLIN	GALSOS
remove spikes	ADESPIKE	ADESPIKE	ADESPIKE	GLLFILLIN
radiometric correction	FICOR77	GALSOS	FICOR77	ADESPIKE

Figure 11: VICAR routine comparisons

However, determining the correct ordering of actions can sometimes be complex. In this case, the correct order to achieve the goals of line fillin, spike removal, and radiometric correction is dependent upon the project of the image file. In the case of Voyager files, ADESPIKE (spike removal) requires raw pixel values and FICOR77 (radiometric correction) changes pixel values to correct for camera response function -- thus FICOR77 removes a necessary condition for ADESPIKE (raw pixel values). This interaction can be avoided by enforcing that ADESPIKE occurs before FICOR77. Additionally, VGRFILLIN requires binary EDR header on the image file, and ADESPIKE removes the binary EDR header, thus ADESPIKE removes a necessary condition for VGRFILLIN. This interaction can be avoided by requiring VGRFILLIN to be executed before ADESPIKE. Thus in the VOYAGER example the only legal execution order is VGRFILLIN, ADESPIKE, FICOR77.

In the Galileo case, GALSOS undoes missing line fillin (the goal achieved by the GLLFILLIN operator). Thus in order to avoid undoing this processing, GLLFILLIN must be applied after GALSOS. Additionally, GALSOS requires raw pixel values, and ADESPIKE alters the pixel values, so ADESPIKE removes a necessary condition for GALSOS. This interaction can be avoided by requiring that GALSOS occur before ADESPIKE.

This simple example, depicted in Figure 11, illustrates some of the interactions and context-sensitivity of the VICAR image processing application. All of these interactions and context sensitive requirements are derived and accounted for automatically by MVP using the operator specification, thereby allowing plan construction despite the presence of complex interactions and conditions.

Impact of the MVP system

The MVP system was deployed to the Multimission Image Processing Laboratory at JPL in 1994. Since then it has been in use by analysts assisting in producing certain classes of science data products. User reports indicate that MVP reduces effort to generate an initial PDF for an expert analyst from 1/2 a day to 15 minutes and reduces the effort for a novice analyst from several days to 1 hour. This represents over an order of magnitude in speedup. The analysts also judged that the quality of the PDFs produced using MVP are comparable to the quality of completely manually derived PDFs.

The Automated SAR Image Processing (ASIP) System

ASIP automates synthetic aperture radar (SAR) image processing based on user request and a knowledge-base model of SAR image processing using AI automated planning techniques [14, 15]. SAR operates simultaneously in multipolarizations and multifrequencies to produce different images consisting of radar backscatter coefficients (s0) through different polarizations at different frequencies. ASIP enables construction of an aerodynamic roughness image/map ($z0^9$ map) from this raw data - thus enabling studies of Aeolian processes.

[9] z0 is pronounced "z-naught" referring to the "z" axis (elevation) and a zero (naught) velosity.

Studies of Aeolian Processes

The aerodynamic roughness length (z0) is the height above a surface at which a wind profile assumes zero velocity. z0 is an important parameter in studies of atmospheric circulation and aeolian sediment transport (in laymans terms: wind patterns, wind erosion patterns, and sand/soil drift caused by wind) [18, 19, 20] Estimating z0 with radar is very beneficial because then large areas can be mapped quickly to study aeolian processes, as opposed to the slow painstaking process of manually taking field measurements [1]. The final science product is a VICAR image called a z0 map that the scientists use to study the aeolian processes.

Figure 12: Aerodynamic Roughness Length Map Produced Using ASIP

Planning to Generate Aerodynamic Roughness Maps

ASIP is an end-to-end image processing system automating data abstraction, decompression, and (radar) image processing sub-systems, and intelligently integrates a number of SAR and z0 image processing sub-systems. Using a knowledge base of SAR processing actions and a general-purpose planning engine, ASIP reasons about the parameter and sub-system constraints and requirements. In this fashion ASIP extracts needed parameters from image format and header files as appropriate, relieving the user of having to know about these aspects of the problem. These parameters, in conjunction with the knowledge-base of SAR processing steps, and a minimal set of required user inputs (entered through a single graphical user interface (GUI)), are then used to create the processing plan. ASIP represents a number of processing constraints. For example, ASIP represents the fact that only some subset of all possible combinations of polarizations are legal (as dependent on the input data). ASIP also represents image processing knowledge about how to use polarization and frequency band information to compute parameters used for later processing of backscatter to aerodynamic roughness length conversion - thus freeing the user from having to understand these processes.

For example, in generating an aerodynamic roughness length map, the user must perform several steps: (1) data acquisition - getting the data from a proprietary tape format using the CEOS reader software package; (2) data conversion: the data must be decompressed by using another software package; (3) pre-processing: header and label files must be added to the data files; (4) processing: using the z0map software package the actual z0 map can be constructed; and (5) post processing: depending on the desired output the z0 image may need to be converted to other proprietary formats for subsequent processing.

Figure 12 shows an aerodynamic roughness length map of a site near Death Valley, California generated using the ASIP system (the map uses the L band (24 cm) SAR with HV polarization). Each of the greyscale bands indicated signifies a different approximate aerodynamic roughness length. This map is then used to study aeolian processes at the Death Valley site.

Since the ASIP system has been fielded, it has proven to be very useful in the use of generating aerodynamic roughness maps with three major benefits. First, ASIP has enabled a 10-fold reduction in the

number of manual inputs required to produce an aerodynamic roughness map. Second, ASIP has enabled a 30% reduction in CPU processing time to produce such a map. Third, and most significantly ASIP has enabled scientists to process their own data (previously programming staff were required). By enabling scientists to directly manipulate that data and reducing processing overhead and turnaround, science is directly enhanced.

Related Work

Related work can be broadly classified into: related image processing languages, related automated image processing work, and related AI planning work. In terms of related image processing languages, there are many commercial and academic image processing packages - such as IDL, Aoips, and Merlyn. Generally, these packages have only limited ability to automatically determine how to use different image processing programs or algorithms based on the problem context (e.g., other image processing goals and initial image state). These packages only support such context sensitivity for a few pre-anticipated cases.

However, there are several previous systems for automatic image processing that use a domain independent mechanism. Work at the Canadian Centre for Remote Sensing (CCRS) [4] has been towards a case-based system for image processing and acquisition of image processing knowledge. This work differs from MVP and ASIP in that they use a case-based reasoning approach in which an existing image processing problem is solved by retrieving a previous problem and solution and adapting it to solve the current problem. Grimm and Bunke [16] developed an expert system to assist in image processing within the SPIDER library of image processing routines. This system uses many similar approaches in that: 1. it classifies problem types similar to the fashion in, which MVP (and ASIP) performs skeletal planning; and 2. it also decomposes larger problems into subproblems which MVP (and ASIP) performs in decomposition planning. This system is implemented in a combination of an expert system shell called TWAICE (which includes both rules and frames) and Prolog. This very basic implementation language gives them considerable power and flexibility but means that their overall system uses a less declarative representation than our decomposition rules and operators which have a strict semantics [12,3]. Previous work on automating the use of the SPIDER library includes [28], which performs constraint checking and step ordering for a set of conceptual image processing steps and generation of executable code. This work differs from MVP/ASIP in that: 1. they do not infer missing steps from step requirements; 2. they do not map from a single abstract step to a context-dependent sequence of image processing operations; and 3. they do not reason about negative interactions between subproblems. MVP/ASIP has the capability to represent and reason about all 3 of these cases. Other work by Jiang and Bunke [21] involves generation of image processing procedures for robotics. This system performs subgoaling to construct image processing plans. However their algorithm does not appear to have a general way of representing and dealing with negative interactions between different subparts of the plans. In contrast, the general Artificial Intelligence Planning techniques used by MVP use conflict resolution methods to guarantee correct handling of subproblem interactions.

Other work by Zmuda [31] describes work in automatically deriving classification software by using machine learning techniques. However for the MVP applications, the search space of possible programs is too large and there is no end feedback (as in classification) to drive the learning process. Another piece of related work is the SATI system [2], which uses an interactive dialogue with the user to drive an automated programming approach to generating code to satisfy the user request. OCAPI [11] a semantically integrated automated image processing system, while being very general provides no clear way to represent the large number of logical constraints associated with the problems MVP/ASIP was designed to solve. Another image processing system [24] provides a means for representing knowledge of image analysis strategies in an expert system but does not use the more declarative AI planning representation. Perhaps the most similar planning and image processing system is COLLAGE [22]. The COLLAGE planning differs from MVP and ASIP in that COLLAGE uses solely the decomposition approach to planning. COLLAGE differs from MVP in the applications sense in that it focuses primarily on earth imaging applications in the Khoros environment, where MVP has focused on planetary applications in the VICAR environment.

Other related work in automatic image processing focuses on speeding execution of algorithms [25, 27] through parallelism but requires that the image processing plans be manually constructed into task

networks whereas MVP automatically constructs the task network from the goal specification and initial image state information.

From the standpoint of planning technology, MVP and ASIP differ from other planning work in two ways. First, it integrates decomposition-based (also called hierarchical task network) and operator-based approaches to more closely model how human experts solve image processing problems. Second, it uses an explicit constraint model to efficiently search among operator effects (which correspond to VICAR program options).

Conclusions

This paper has described knowledge-based reconfiguration of data analysis software using AI planning techniques. This represents an important area where AI planning can significantly enhance KDD processes. As evidence of this potential, we described two fielded planning systems that enhance KDD: the MVP system, which automates image processing to support Galileo image data science analysis; and the ASIP system which automates production of aerodynamic roughness maps to support geological science analysis.

Acknowledgements

This work was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The authors would also like to acknowledge the contributions of other past and present members of the MVP team: Todd Turco, Christine Ying, Shouyi Hsiao, Darren Mutz, Alex Gray, Joe Nieten, and Jean Lorre. The authors would also like to acknowledge other contributors to the ASIP project including: Dan Blumberg(ASU), Anita Govindjee, John McHone(ASU), Keld Rasmussen(ASU), and Todd Turco.

Bibliography

1. Blumberg, D. and Greeley, R., Field Studies of Aerodynamic Roughness Length, *Journal of Arid Environments.* 25, 39-48, 1993.

2. Capdevielle, and Dalle, P., Image Processing Chain Construction by Interactive Goal Specification, in Proceedings of the *First IEEE International Conference on Image Processing,* Austin, TX, Vol. 3, 816-819, 1994.

3. Chapman, D., Planning for Conjunctive Goals, *Artificial Intelligence.* 32 , 3, 1987.

4. D. Charlebois, J. DeGuise, G. Goodenough, S. Matwin, and M. Robson, A Case-based Planner to Automate Reuse of ES Software for Analysis of Remote Sensing Data, *International Geoscience and Remote Sensing Symposium* (IGARSS), Vol 3, pp. 1851-1854, 1991.

5. S. A. Chien and H. B. Mortensen, Automating Image Processing for Scientific Data Analysis of a Large Image Database, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18 (8): pp. 854-859, August 1996.

6. S. Chien, D. DeCoste, R. Doyle, and P. Stolorz, Making an Impact: Artificial Intelligence at the Jet Propulsion Laboratory, *AI Magazine* 18 (1): Spring 1997, pp. 103-122.

7. S. Chien, A. Govindjee, T. Estlin, X. Wang, A. Griesel, R. Hill Jr., Automated Generation of Tracking Plans for a Network of Communications Antennas, The Proceedings of the. 1997 IEEE Aerospace Conference, Aspen, CO, February, 1997.

8. S. Chien, F. Fisher, E. Lo, H. Mortensen, R. Greeley, Using Artificial Intelligence Planning to Automate Science Data Analysis for Large Image Databases, *The Proceedings of the 1997 Conference on Knowledge Discovery and Data Mining,* Newport Beach, CA, August 1997.

9. S. Chien, N. Muscettola, K. Rajan, B. Smith, G. Rabideau, Automated Planning and Scheduling for Goal-based Autonomous Spacecraft, *IEEE Intelligent Systems,* September/October 1998, pp. 50-55.

10. S. Chien, F. Fisher, H. Mortensen, E. Lo, R. Greeley, A. Govindjee, T. Estlin, X. Wang, Using Artificial Intelligence Planning Techniques to Automatically Reconfigure Software Modules, *11th International Conference on Industrial and Engineering Application of Artificial Intelligence and Expert Systems*, Castellon Spain, June 1998.

11. V. Clement and M. Thonnat, A Knowledge-based Approach to Integration of Image Processing Procedures, *Image Understanding,* 57:166-184, March 1993.

12. K. Erol, J. Hendler, and D. Nau, UMCP: A Sound and Complete Procedure for Hierarchical Task Network Planning, *Proceedings of the 2nd International Conference on AI Planning Systems,* Chicago, IL, June 1994, pp. 249-254.

13. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From Data Mining to Knowledge Discovery in Databases, *AI Magazine*, Vol 17 No. 3, Fall 1996, pp. 37-54.

14. F. Fisher, E. Lo, S. Chien, R. Greeley, SAR Image Processing Through Artificial Intelligence Planning, *Proceedings of the IEEE Aerospace Conference*, Aspen, CO, March 1999.

15. F. Fisher, S. Chien, E. Lo, R. Greeley, Using Artificial Intelligence Planning to Automate SAR Image Processing for Scientific Data Analysis, *Proceedings of the 1998 Conference on Innovative Application of Artificial Intelligence*, Madision, WI, July 1998.

16. F. Grimm and H. Bunke, An Expert System for the Selection and Application of Image Processing Subroutines, *Expert Systems,* May 1993, Vol. 10, No. 2, pp. 61-74.

17. Y. Iwasaki and P. Friedland, The Concept and Implementation of Skeletal Plans, *Journal of Automated Reasoning* 1, 1 (1985), pp. 161-208.

18. R. Greeley and J.D. Iversen, Measurements of Wind Friction Speeds over Lava Surfaces and Assessment of Sediment Transport, *Geophysical Research Letters.* 14 (1987):925-928.

19. R. Greeley, P.R. Christensen, and J.F. McHone, Radar Characteristics of Small Craters: Implications for Venus, *Earth, Moon and Planets* 37 (1987):89-111.

20. R. Greeley, L. Gaddis, A. Dobrovolskis, J. Iversen, K. Rasmussen, S. Saunders, J. vanZyl, S. Wall, H. Zebker, and B. White, Assessment of Aerodynamic Roughness Via Airborne Radar Observations, 1991, *Acta Mechanica Suppl.*2, 77-88.

21. X. Jiang and H. Bunke, Vision Planner for an Intelligence Multisensory Vision System, *Technical Report*, University of Bern (extended version of a paper appearing in *ICPR* 1994).

22. A. Lansky, M. Friedman, L. Getoor, S. Schmidler, and N. Short Jr., The Collage/Khoros Link: Planning for Image Processing Tasks, Proceedings of the 1995 *AAAI Spring Symposium on Integrated Planning Applications*, March 1995, pp. 67-76.

23. S. LaVoie, D. Alexander, C. Avis, H. Mortensen, C. Stanley, and L. Wainio, VICAR User's Guide, Version 2, *JPL Internal Document* D-4186, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 1989.

24. T. Matsuyama, Expert Systems for Image Processing: Knowledge-Based Composition of Image Analysis Processes, *Computer Vision, Graphics, and Image Processing* 48, (1989), pp. 22-49.

25. M. Moore, G. Karsai, and J. Sztipanovits, Model-based Programming for Parallel Image Processing, Proceedings of the First *IEEE International Conference on Image Processing,* Austin, TX, Nov 1994, Vol. 3, pp. 811-815.

26. J. S. Pemberthy and D. S. Weld, UCPOP: A Sound Complete, Partial Order Planner for ADL, *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning,* Oct 1992.

27. P. Romig, A. Samal, Devious: A Distributed Environment for Vision Tasks, Proceedings of the First *IEEE International Conference on Image Processing,* Austin, TX, Nov 1994, Vol. 3, pp. 786-790.

28. K. Sakaue and H. Tamura, Automatic Generation of Image Processing Programs by Knowledge-based Verification, *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 189-192, 1985.

29. D. Weld, An introduction to least commitment planning, *AI Magazine* 15(4):27-61, 1994.

30. D. Wilkins, Practical Planning: Extending the Classical AI Planning Paradigm, Morgan Kaufmann, 1988.

31. M. Zmuda, M. Rizki, and L. Tamburino, Approaches to Synthesizing Image Processing Programs, *IEEE National Aerospace and Electronics Conference*, Vol. 3, pp. 1054-1059, 1991.